

2FW

00862.023513

PATENT APPLICATION



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:)	
	:	Examiner: Not Yet Assigned
AKIHIRO MITSUI \)	
	:	Group Art Unit: NYA
Application No.: 10/807,102)	
	:	
Filed: March 24, 2004)	
	:	
For: INFORMATION PROCESSING)	
APPARATUS AND METHOD	:	May 19, 2004

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

SUBMISSION OF PRIORITY DOCUMENTS


Sir:

In support of Applicant's claim for priority under 35 U.S.C. § 119, enclosed are
certified copies of the following Japanese applications:

- 2003-089139, filed March 27, 2003; and
- 2004-074706, filed March 16, 2004.

Applicant's undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All correspondence should continue to be directed to our address given below.

Respectfully submitted,



Attorney for Applicant

Registration No. 29,286

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3800
Facsimile: (212) 218-2200

NY_MAIN 428653v1

日本国特許庁
JAPAN PATENT OFFICE

10/807,102

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: 2004年 3月16日

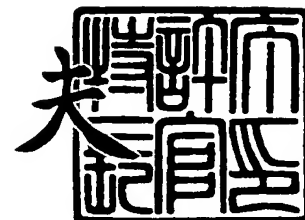
出願番号
Application Number: 特願2004-074706
[ST. 10/C]: [JP 2004-074706]

出願人
Applicant(s): キヤノン株式会社

2004年 4月12日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫



【書類名】 特許願
【整理番号】 0001445-01
【提出日】 平成16年 3月16日
【あて先】 特許庁長官殿
【国際特許分類】 G06F 15/00
【発明者】
 【住所又は居所】 東京都大田区下丸子 3 丁目 3 0 番 2 号 キヤノン株式会社内
 【氏名】 三井 章弘
【特許出願人】
 【識別番号】 000001007
 【氏名又は名称】 キヤノン株式会社
【代理人】
 【識別番号】 100076428
 【弁理士】
 【氏名又は名称】 大塚 康德
 【電話番号】 03-5276-3241
【選任した代理人】
 【識別番号】 100112508
 【弁理士】
 【氏名又は名称】 高柳 司郎
 【電話番号】 03-5276-3241
【選任した代理人】
 【識別番号】 100115071
 【弁理士】
 【氏名又は名称】 大塚 康弘
 【電話番号】 03-5276-3241
【選任した代理人】
 【識別番号】 100116894
 【弁理士】
 【氏名又は名称】 木村 秀二
 【電話番号】 03-5276-3241
【先の出願に基づく優先権主張】
 【出願番号】 特願2003- 89139
 【出願日】 平成15年 3月27日
【手数料の表示】
 【予納台帳番号】 003458
 【納付金額】 21,000円
【提出物件の目録】
 【物件名】 特許請求の範囲 1
 【物件名】 明細書 1
 【物件名】 図面 1
 【物件名】 要約書 1
 【包括委任状番号】 0102485

【書類名】 特許請求の範囲**【請求項 1】**

デバイスドライバに設定される設定値に従って周辺装置を制御する情報処理装置であって、

前記デバイスドライバの設定値間の不整合を回避するための、複数のコンフリクト処理ルールを保持する保持手段と、

前記複数のコンフリクト処理ルールに従って、対応する設定値の状態を制御する制御手段と、

を有し、

前記複数のコンフリクト処理ルールの少なくともいずれかは、他のコンフリクト処理ルールに対する適用優先度の情報を含むことを特徴とする情報処理装置。

【請求項 2】

前記制御手段は、前記適用優先度の情報に応じた順序で各コンフリクト処理ルールを適用することを特徴とする請求項 1 に記載の情報処理装置。

【請求項 3】

前記適用優先度の情報は、特定の状態変数に係る後続のコンフリクト処理ルールを優先適用するよう指示する制御情報として記述され、

前記制御手段は、読み出したコンフリクト処理ルールから前記制御情報を抽出し、当該制御情報に基づき優先適用すべき他のコンフリクト処理ルールを読み出し、該他のコンフリクト処理ルールを評価することを特徴とする請求項 2 に記載の情報処理装置。

【請求項 4】

前記デバイスドライバの設定値は、印刷方法として両面印刷又は製本印刷の指定情報、および、印刷データのスケーリング機能の有効／無効を示す ON / OFF の情報を含み、

前記複数のコンフリクト処理ルールは、印刷方法として両面印刷又は製本印刷が指定されている場合は印刷データのスケーリング機能を OFF に設定する、という論理を含むことを特徴とする請求項 1 から 3 までのいずれかに記載の情報処理装置。

【請求項 5】

前記優先適用度の情報は、印刷方法の指定に係るコンフリクト処理ルールを優先して適用するよう指示する制御情報として、印刷データのスケーリング機能に係るコンフリクト処理ルール中に記述されることを特徴とする請求項 4 に記載の情報処理装置。

【請求項 6】

デバイスドライバに設定される設定値に従って周辺装置を制御するための情報処理方法であって、

前記デバイスドライバの設定値間の不整合を回避するための、複数のコンフリクト処理ルールを読み込む読み込みステップと、

読み込んだ前記複数のコンフリクト処理ルールに従って、対応する設定値の状態を制御する制御ステップと、

を有し、

前記複数のコンフリクト処理ルールの少なくともいずれかは、他のコンフリクト処理ルールに対する適用優先度の情報を含むことを特徴とする情報処理方法。

【請求項 7】

前記制御ステップは、前記適用優先度の情報に応じた順序で各コンフリクト処理ルールを適用することを特徴とする請求項 6 に記載の情報処理方法。

【請求項 8】

前記適用優先度の情報は、特定の状態変数に係る後続のコンフリクト処理ルールを優先適用するよう指示する制御情報として記述され、

前記制御ステップは、読み出したコンフリクト処理ルールから前記制御情報を抽出し、当該制御情報に基づき優先適用すべき他のコンフリクト処理ルールを読み出し、該他のコンフリクト処理ルールを評価することを特徴とする請求項 7 に記載の情報処理方法。

【請求項 9】

前記デバイスドライバの設定値は、印刷方法として両面印刷又は製本印刷の指定情報、および、印刷データのスケーリング機能の有効／無効を示す O N / O F F の情報を含み、

前記複数のコンフリクト処理ルールは、印刷方法として両面印刷又は製本印刷が指定されている場合は印刷データのスケーリング機能を O F F に設定する、という論理を含むことを特徴とする請求項 6 から 8 までのいずれかに記載の情報処理方法。

【請求項 1 0】

前記優先適用度の情報は、印刷方法の指定に係るコンフリクト処理ルールを優先して適用するよう指示する制御情報として、印刷データのスケーリング機能に係るコンフリクト処理ルール中に記述されることを特徴とする請求項 9 に記載の情報処理方法。

【請求項 1 1】

デバイスドライバに設定される設定値に従って周辺装置を制御するための、コンピュータによって実行されるプログラムであって、

前記デバイスドライバの設定値間の不整合を回避するための、複数のコンフリクト処理ルールを読み込んでメモリに格納するためのコードと、

読み込んだ前記複数のコンフリクト処理ルールに従って、対応する設定値の状態を制御するためのコードと、

を含み、

前記複数のコンフリクト処理ルールの少なくともいずれかは、他のコンフリクト処理ルールに対する適用優先度の情報を含む

ことを特徴とするプログラム。

【請求項 1 2】

デバイスドライバに設定される設定値に従って周辺装置を制御するための、コンピュータによって実行されるプログラムを格納した記憶媒体であって、前記プログラムは、

前記デバイスドライバの設定値間の不整合を回避するための、複数のコンフリクト処理ルールを読み込んでメモリに格納するためのコードと、

読み込んだ前記複数のコンフリクト処理ルールに従って、対応する設定値の状態を制御するためのコードと、

を含み、

前記複数のコンフリクト処理ルールの少なくともいずれかは、他のコンフリクト処理ルールに対する適用優先度の情報を含む

ことを特徴とする記憶媒体。

【書類名】明細書

【発明の名称】情報処理装置および方法

【技術分野】

【0001】

本発明は、情報処理装置および方法に関し、特に、デバイスドライバに設定される設定値に従って周辺装置を制御する情報処理装置および方法に関する。

【背景技術】

【0002】

ユーザインターフェース（以下、「UI」ともいう。）を介して、ユーザから複数の設定値の入力を受け付け、それらの設定値に基づき制御される機器は、数多くある。ホストコンピュータに接続され、そのホストコンピュータにおいて設定された情報に基づき画像形成処理を行う画像形成システム（プリンタシステム）はその一例である。プリンタシステムのホストコンピュータは一般に、印刷動作を制御するいわゆるプリンタドライバ、およびユーザからの印刷設定などを受け付けるUIを含む印刷関連処理プログラムを有する。

【0003】

印刷関連処理プログラムは、UIを介してユーザからの設定値の入力を受け付ける度に、それまでに設定された複数の設定値の中で関連する設定値の値との関係性を評価し、設定値間の不整合（コンフリクト）がないかどうかを判別する。コンフリクトの例としては、記録媒体としてセットされたOHPシートに対して両面印刷を行わせるような設定などのユーザにとって不都合と予想される設定や、プリンタに不可能な動作を行わせる設定などがある。

【0004】

したがって、コンフリクトが存在した場合には、それを回避するためのコンフリクト処理が必要になる。

【0005】

コンフリクト処理の実現にあたっては、コンフリクトの検知および解消処理を設定値の関係性に依存した形で専用のプログラムを用いて記述する、もしくはコンフリクト処理が必要となる複数の設定値の条件を一覧としてファイルに保存し、このファイルをコンフリクト処理プログラムに読み込ませることで、コンフリクト処理プログラムが汎用的に利用できるようにしていたかのどちらかであった。

【0006】

【特許文献1】特開2002-202865号公報

【特許文献2】特開2003-296063号公報

【特許文献3】特開2002-169669号公報

【特許文献4】特開2002-202868号公報

【特許文献5】特開2002-328787号公報

【特許文献6】特開2002-328757号公報

【特許文献7】特開2003-099170号公報

【特許文献8】特開2002-196905号公報

【特許文献9】特開2003-091373号公報

【特許文献10】特開2004-005194号公報

【発明の開示】

【発明が解決しようとする課題】

【0007】

ファイルにコンフリクト処理を保存する際の記述に関しては、設定値間の項目に関するルールを記述するに過ぎず、複数のルールが記述されていたときには、コンフリクト処理とは直接関係のない記述順によって順番に処理されていた。また、コンフリクト処理によって変更された設定値がさらにコンフリクト処理によって変更されるときに、どのルールから設定値が定まっていくのかが明確に定義できないため、記述を行っては実行しコンフ

リクト処理プログラムが解釈する順番を手探りで探していかなければならず、誤判断や工数の増加につながっていた。

【0008】

そこで、本発明は、開発者のミスや作業工数を減らすことができるよう、ユーザインターフェースのコンフリクト処理ルールの記述構造を改良することを目的とする。

【課題を解決するための手段】

【0009】

上記した課題は、本発明の情報処理装置および方法によって解決される。本発明の一側面は、デバイスドライバに設定される設定値に従って周辺装置を制御する情報処理装置に係り、前記デバイスドライバの設定値間の不整合を回避するための、複数のコンフリクト処理ルールを保持する保持手段と、前記複数のコンフリクト処理ルールに従って、対応する設定値の状態を制御する制御手段とを有し、前記複数のコンフリクト処理ルールの少なくともいずれかは、他のコンフリクト処理ルールに対する適用優先度の情報を含むことを特徴とする。

【0010】

また、本発明の別の側面は、デバイスドライバに設定される設定値に従って周辺装置を制御するための情報処理方法に係り、前記デバイスドライバの設定値間の不整合を回避するための、複数のコンフリクト処理ルールを読み込む読み込みステップと、前記複数のコンフリクト処理ルールに従って、対応する設定値の状態を制御する制御ステップとを有し、前記複数のコンフリクト処理ルールの少なくともいずれかは、他のコンフリクト処理ルールに対する適用優先度の情報を含むことを特徴とする。

【発明の効果】

【0011】

本発明によれば、ユーザインターフェースのコンフリクト処理ルールの記述構造が改良され、これにより開発者のミスや作業工数の削減に貢献することができる。

【発明を実施するための最良の形態】

【0012】

以下、図面を参照して本発明の好適な実施形態について詳細に説明する。

【0013】

図1は本発明の一実施形態における印刷処理システムのブロック構成図である。なお、特に断らない限り、本発明の機能が実行されるのであれば、単体の機能であっても、複数の機器からなるシステムであっても、LAN、WAN等のネットワークを介して接続がなされ処理が行われるシステムであっても、本発明を適用できることは言うまでもない。

【0014】

図1は、一般的なコンピュータを用いた印刷処理システムの例を示している。CPU100はROM101あるいはRAM102あるいは外部記憶装置107に格納されたプログラムに従って装置全体の制御を行う。RAM102はCPU100が各種処理を行う際のワークエリアとして使用される。

【0015】

ハードディスク装置などで実現される外部記憶装置107は、オペレーティングシステム(OS)1071やアプリケーションソフト1072、および印刷関連処理プログラム1073を記憶している。ここで、印刷関連処理プログラム1073は、デバイスドライバとしてのプリンタドライバ1074およびプリンタドライバUI制御モジュール1075を含む構成である。

【0016】

キーボード103もしくは非図示のマウスなどの入力機器は、ユーザーが各種指示を与えるためのデバイスである。ネットワークI/F1041はLAN104もしくは非図示のネットワークに接続しデータの授受を行うためのインターフェースである。モニタI/F1051はモニタ105に接続しデータの授受を行うためのインターフェースである。プリンタI/F1061は、被制御装置であるプリンタ106と接続しデータの授受を行

うためのインターフェースである。また、108は共通データベースである。なお、プリンタ106は被制御装置の一例であり、本発明はスキャナ等を被制御装置としても適用可能である。

【0017】

図2は、実施形態における印刷関連処理プログラム1073のプリンタドライバUI制御モジュール1075の機能構成を示している。203は、各モジュール間のデータの受け渡しやデータの更新等を管理してコンフリクト処理を統括するコンフリクトマネージャである。206が、印刷設定画面の表示制御を行うプリンタドライバUIである。201は、プリンタドライバUI206を介して入力される設定値間の不整合を回避するための、複数のコンフリクト処理ルールを記述したコンフリクト処理ルール記述ファイルである。202は、コンフリクト処理ルール記述ファイル201をロードして、入力された設定値に対してコンフリクト処理ルールを適用し、各機能の状態を推論する推論エンジン、204は、各プリンタ機能の状態をリスト形式で表示する状態変数リストであり、ユーザからの入力およびコンフリクト処理ルール記述ファイル201の内容に基づき更新されうる。205は、プリンタドライバUI206が提供する画面表示の基になる帳票としての内部構造体であり、状態変数リスト204と連動して各プリンタ機能の状態を所定の形式で表示する。

【0018】

プリンタドライバUI206を介してユーザからの設定情報を受け取ったコンフリクトマネージャ203は、コンフリクト処理ルール記述ファイル201を参照する。このことは、図示の如くコンフリクト処理ルール記述ファイル201からコンフリクトマネージャ203に向かう矢印で、「R(Read)」として表示されている。参照の結果、設定情報がコンフリクト処理ルールに適合する場合、コンフリクト処理が適用される。そうして、コンフリクトマネージャ203は、状態変数リスト204および内部構造体205を更新してプリンタドライバUI206に反映させる。この更新作業については、図示の如くコンフリクトマネージャ203が、状態変数リスト204および内部構造体205とそれぞれ双方向矢印で結ばれ、「R/W (Read/Write)」として表示されている。

【0019】

図3は、図2に示した各モジュールで扱われるデータの関連を説明する模式図である。図3において、コンフリクト処理ルール記述ファイル201は、推論エンジン202にインクルード（ロード）されたかたちで参照される。このコンフリクト処理ルール記述ファイル201は、コンフリクトマネージャ203にも参照され、それを受けて状態変数リスト204が変更されることになる。また、内部構造体205と状態変数リスト204とは先に述べたとおり連動表示されるものであるから、互いにマッピングされる関係にある。そしてこの状態がプリンタドライバUI206の制御によってユーザに見えるかたちで表現される。

【0020】

図4は、実施形態におけるコンフリクト処理ルールの適用手順を示すフローチャートである。ここでは一例として、プリンタ106を制御するためのプリンタドライバ1074によってモニタ105に各種設定用に表示されるプリンタドライバUI上で行われるコンフリクト処理を説明する。

【0021】

図4に示す処理は、OSもしくはアプリケーション上で、ユーザがキーボード103などによりプリンタドライバUIを開く指示をすることで始まる。ユーザによるプリンタドライバUIを開く指示により、OS1071の管理の下、RAM102に印刷関連処理プログラム1073がロードされる。印刷関連処理プログラムがRAM102にロードされると、まずプリンタドライバUIを開くための初期化処理が行われる。推論エンジン202は、図6に例示するような表記のコンフリクト処理ルールを記述したファイル201をコンフリクトマネージャ203を介して、RAM102の中に読み込む（ステップS401）。

【0022】

コンフリクト処理ルール記述ファイル201に記述される各ルールは、論理（ロジック）を用いて数学的に形式化される。述語は、「プリンタ機能名（引数）」の形で記述される。引数としては、ON/OFFの他、数値が入る場合もある（例えば、印刷部数等）。本明細書では、このような引数の内容によって現在の状態が表現されうる機能を「状態変数」とよぶ。つまり、各ルールは「状態変数」を用いた論理式で記述される。具体的には、左辺には「プリンタ機能名（引数）」を、右辺には左辺が成り立つための論理条件を記述し、記号「:-」で関係づける。例えば、

A(ON) :- B(ON).

は、「プリンタ機能Bの状態がONのときは、プリンタ機能Aの状態をONとする」という意味のルールになる。

【0023】

また、式中の記号「,」は「かつ（AND）」の意味で用いられる。例えば、「プリンタ機能Bの状態がON、かつ、プリンタ機能Cの状態がOFFのとき、プリンタ機能Aの状態をONとする」というルールは、

A(ON) :- B(ON), C(OFF).

と記述される。

【0024】

ここで例えば、コンフリクト処理ルール記述ファイル201に、次のようなコンフリクト処理ルールが記述されていたとする。

A(ON) :- B(ON), C(OFF).

【0025】

状態変数リスト204には、図3に示すように、コンフリクト処理ルール記述ファイル201に記述されたプリンタの機能名A, B, Cがそれぞれ同名の状態変数として同図中の「キー」欄に記述される。また、各状態変数の現在値（ON/OFFなど）が、「値」欄に記述される。ここでは、プリンタ機能名A, B, Cに対応するプリンタドライバUIの内部構造体205のメンバをそれぞれ a, b, c で表現する。内部構造体205は内部もしくは外部に初期値をもっており、状態変数の値は対応する内部構造体205のメンバの初期値に依存する。aの初期値は0なので、対応する状態変数Aの値はOFFとなっている。

【0026】

また、状態変数リスト204は、図3に示すように、後述する各ルールの適用順序に関する遅延フラグと遅延情報を持ち、これらによってそれぞれ自分自身が遅延情報を持つかどうか、他の遅延情報を持つかどうか保存される。詳細は後述する。

【0027】

推論エンジン202はコンフリクトマネージャ203からコンフリクト処理ルール記述ファイル201を渡され、状態変数の推論を行ってコンフリクト処理ルールの評価を行う。状態変数Bの値がONであり、かつ、状態変数CがOFFであったときは上記のコンフリクト処理ルールの右辺の評価が成立し、左辺の状態変数Aの値をONに変更する。コンフリクト処理ルールの推論が終了すると、コンフリクトマネージャ203は変更された状態変数の値をプリンタドライバUIの内部構造体205の対応するメンバaに反映させる。つまり、状態変数Aの値がOFFからONになったことで、内部構造体205のメンバaは上記のコンフリクト処理ルールが成立したことによって、0から1に変更される。

【0028】

コンフリクト処理ルールでは、プリンタの機能と結び付けられた状態変数のほかに、一時的に値を記憶しておくための状態変数を記述することもできる。一時的に値を記憶しておくための状態変数は推論エンジン202がコンフリクト処理ルールに従いコンフリクト処理を行い始めてから、コンフリクト処理を終了するまで、推論エンジン202の内部に値が保存される。一時的な状態変数なので、コンフリクト処理が終了すれば消える揮発性のデータとなる。

【0029】

一時的な状態変数の記述の方法は、状態変数リスト204に持たないキー名称をコンフリクト処理ルール内で状態変数として記述しておく、推論エンジン202はこれを一時的な状態変数とみなして処理を行う。一時的な状態変数は初期の値を定めることができないので、作成された段階での初期の値は不定値となる。

【0030】

一時的な状態変数は、複雑なコンフリクト処理の最中で、途中のコンフリクト処理の状態を覚えておきたいときや、1対多のコンフリクト処理ルールの対応付けをとるのに利用することができる。

【0031】

ここで、コンフリクト処理ルール記述ファイル201に、図6のようなコンフリクト処理ルールが記述されており、これをコンフリクトマネージャ203が推論エンジン202に渡すとする。推論エンジン202はコンフリクト処理ルール記述ファイル201を、記述された順に従って自身の持つ待ち行列に追加し、図9のフローチャートに従って評価を開始する。

【0032】

推論エンジン202は自身の待ち行列の先頭から、コンフリクト処理ルールを取り出す(ステップS901)。すなわち、まず図6の601で示されるルール

A(ON) :- B(OFF), wait([C]).

が取り出される。

【0033】

次に、取り出したルールの右辺に制御記号「wait」が含まれているか否かを判断する(ステップS902)。この制御記号「wait」は、この制御記号が記述されているコンフリクト処理ルールの、他のコンフリクト処理ルールに対する適用優先度の情報を表す。601のルールにはこの「wait」が含まれている。このように右辺に「wait」の記号があったときは、図3に示すように、「wait」で指定された状態変数Cに対し状態変数リスト204における遅延フラグを「TRUE」とするとともに、左辺に対応する状態変数Aに対し遅延情報として「C」を記述する(ステップS903)。これにより状態変数Aは状態変数Cが先に評価されるのを待っていることが示される。

【0034】

次に、右辺から「wait」を取り除いたルールを生成する(ステップS904)。すなわち、601のルールから、

A(ON) :- B(OFF). (603)

が生成される。そして、この603のルールを待ち行列の最後に挿入する(ステップS905)。

【0035】

その後、ステップS901に戻って、次のコンフリクト処理ルール、すなわち、602で示されるコンフリクト処理ルール

B(OFF) :- C(ON).

を待ち行列から取り出し、ステップS902で、右辺に「wait」の記号が含まれているか否かを判断する。この場合には右辺に「wait」は存在しないので、ステップS906に進み、右辺の評価を行う。推論エンジン202は、左辺の状態変数Bにおける遅延情報が状態変数Cに結び付けられているかどうかを判定する(ステップS907)。しかしこの例では、602のルールの左辺にある状態変数Bには状態変数Cに関する遅延情報はないので、そのままステップS910に進み、式の評価を行う。推論エンジン202は式を評価し、602のルールに従い、状態変数Cの値をコンフリクトマネージャ203に問い合わせた後、状態変数Cの値がONであれば状態変数Bの値をOFFにする。そして、状態変数Cの遅延フラグを「FALSE」に変更し、状態変数Cに関する処理が終わったことを示す。

【0036】

ステップS911では、待ち行列にルールがあるかどうかをチェックする。ここで待ち行列にルールがなくなればこの処理を終了することになるが、この例では、ステップS905で、601のルールから「wait」を取り除いたルール603が待ち行列の最後に挿入されたので、処理はステップS901に戻り、ルール603を取り出す。この場合、ルール603の右辺に「wait」は存在しないので、ステップS902からステップS906に進み、右辺の評価を開始する。ステップS907では、左辺の状態変数Aにおける遅延情報が状態変数Cに結び付けられているかどうかを判定する。この例では、先ほどのステップS903でルール601の左辺にある状態変数Aの遅延情報が状態変数Cに結び付けられているので、処理はステップS908に進む。

【0037】

ステップS908では、「wait」が付いた状態変数のルールをすべて評価したかどうかを検査する。すなわち、状態変数Aに結び付けられた遅延情報の状態変数Cが評価されたかどうかを確認する。ここで、すべて評価されていないときはステップS905に進むが、この例では、ルール602を評価したステップS910で状態変数Cから遅延フラグが下りていることから、すべて評価されていると判断され、ステップS909に進む。ステップS909では、状態変数Aの遅延情報を削除し、その後、ステップS910で、式を評価する。

【0038】

なお、コンフリクト処理ルールの記述順や評価を行う過程によっては、ステップS908において、「wait」のない状態変数Cの評価がすべて終了したことを判定できない場合があり得る。そのような場合には無限ループに陥ってしまい処理を終了できないことになる。そこで、ステップS910が行われずにステップS901に戻った回数をカウントしておき、そのカウント値が所定回数を超えると、待ち処理をあきらめステップS907からステップS910に強制的に進めるようにすることで、無限ループに陥ることを防止することが好ましい。

【0039】

さて、図5に示したコンフリクト処理ルールの例と図6のコンフリクト処理ルールの例とを比較すると、図5のコンフリクト処理ルールでは、記述順にルールが評価される場合には、状態変数BがONだったときとOFFだったときで、状態変数Aの値が異なってしまうが、図6に示したような「wait」の記述によりルールの適用順序を制御することで、状態変数Aが不定になることを防止できる。

【0040】

以上のような処理によって他に適用されるコンフリクト処理ルールがなくなった時点で、図7に例示するようなプリンタドライバUIをオープンする（ステップ402）。

【0041】

プリンタドライバUIがオープンされた後は、OSより送られてくるイベントの取得とその処理を繰り返す（ステップS403）。ステップS404では、ステップS403にて取得したイベントが、ユーザーがプリンタドライバUI上の設定項目を変更したイベントであるかどうかの判断を行う。ここで、イベントがユーザーの設定項目変更要求であった場合には、その設定項目に対する内部構造体のメンバがコンフリクトマネージャによって変更され、同じくコンフリクトマネージャによって対応づけられた状態変数に変更され、推論エンジンがコンフリクト処理ルールをもとに、上記のコンフリクト処理を行う（ステップS405）。

【0042】

他に適用されるコンフリクト処理ルールがなくなった時点で、コンフリクトマネージャがコンフリクト処理によって変更があったすべての状態変数を、対応する内部構造体のメンバに適用する（ステップS406）。コンフリクトマネージャは変更された内部構造体のメンバから、UIの更新が必要かどうかの判別を行う（ステップ407）。UIの更新が必要ない場合はそのままステップS403に戻るが、更新が必要な場合は、UIの更新処理を行い（ステップS408）、その後ステップS403に戻る。

【0043】

また、ステップS404において、取得したイベントがユーザーの設定項目変更要求でなかった場合には、ステップS409に進み、そのイベントがユーザーによってプリンタドライバUIのクローズ要求かどうかを判別する。ここで、そのイベントがクローズ要求だった場合には、プリンタドライバUIをクローズし終了処理を行って（ステップS410）、本処理を終了し（ステップS411）、そうでなければ、ステップS403に戻って処理を繰り返す。

【0044】

以上の処理は、ユーザーによってプリンタドライバUIがクローズされるまで繰り返し実行される。プリンタドライバUIがクローズされると処理はすべて終了し、本実施形態における印刷関連処理プログラムの処理も終了し、RAM103からはOS1071の機能により消去される。プリンタドライバUIで使用されていた内部構造体205は、プリンタ106もしくはOS1071に反映され、ユーザーがプリンタ106に対して印刷を行う指示をするときに使用される。

【0045】

なお本実施形態においては、本印刷関連処理プログラムを記録する媒体を外部記憶装置107としているが、外部記憶装置は、FD、HDD、光ディスク媒体、ICメモリーカードなどであってもよい。さらに、本印刷関連処理プログラム単独、もしくはOSそのほかコンピュータ上で動作するプログラムとともにROM101に記録しておき、これをメモリマップの一部としてなすように構成し、直接CPU100で実行することも可能である。

【0046】

また、上述した実施形態では、コンフリクト処理ルールの順番待ちを1つのルールを加えることで実現しているが、かわりに、図8に示すように、一連の複数のコンフリクト処理ルールをまとめて、そのひとまとまりの中でのみ互いのルールおよび状態変数が干渉し、それ以外は一切評価を行わないといった意味の、スコープ「**{ }**」、「**[]**」で表現することも可能である。たとえば、推論エンジン202はスコープの始まりを検知すると、いったんスコープ内に記述されたコンフリクト処理ルールを、スコープが現れる順にスコープ番号を添付して順番待ち行列に入れ込む。スコープ内のコンフリクト処理ルールを評価するためには、互いのスコープの順序を記述したコンフリクト処理ルールがあればその順番に従い、存在しないときはスコープの記述順に評価する。コンフリクト処理ルールは、それぞれスコープ番号をもっており、ひとつコンフリクト処理ルールを評価すると、他のスコープ番号のルールが待ち行列から取り出されても、推論エンジンは評価を行わない。これによって、互いに優先順位をつけることができる。

【0047】

以上説明した実施形態によれば、プログラム開発者などが用意するコンフリクト処理ルールで作成されるコンフリクト条件を、推論エンジンに規則的な順序を守って評価をさせることができるので、推論エンジンの動作による結果を表現しやすく、品質の高いコンフリクト処理を実現できるようになる。

【0048】

また、コンフリクト処理ルールの評価が始まる順番によって、結果が変化してしまう従来の問題を防ぐことができる。例えば、図10と図11に示したコンフリクト処理ルールを比べると、両者は同様の内容であるが、図11の第1, 3行のルールには適用順序に係る制御記号「wait」が記述されている。図10の場合、状態変数CもDもONだったときは、状態変数Bが先にONになるかOFFかは評価次第になってしまい、その結果、第1, 3行のルールによってBの状態に依存する状態変数Aも先にONになるかOFFになるかはこれだけでは判定することができない。これはプログラム開発者としては意図した方向にコンフリクト処理を割り振ることができない一例である。一方、図11の記述であれば、第1行のルールは状態変数Cのルールに結びつけられていることが確認でき、また、第3行のルールは状態変数Dのルールに結びつけられていることが確認できるため、意図

した順に状態変数 A, B を変化させることができる。

【 0 0 4 9 】

制御記号「wait」の使用によって、コンフリクト処理ルールの評価が始まる順番により結果が変化してしまう従来の問題が防止される例を以下に示す。コンフリクト処理ルールとして以下のようなルールを作成したとする。

【 0 0 5 0 】

```
$TEMP(OFF) :- Layout$Printing(BOOKLET).           (1)
$TEMP2(OFF) :- $TEMP(OFF).                          (2)
$TEMP3(OFF) :- $TEMP2(OFF).                         (3)
Layout$Scaling(OFF) :- Layout$Printing(TWOSIDED).  (4)
Layout$Scaling(OFF) :- $TEMP3(OFF).                 (5)
```

【 0 0 5 1 】

これは、印刷の方法 (Printing) として製本印刷 (BOOKLET) を選択したとき、もしくは両面印刷 (TWOSIDED) を選択したときには、印刷データのスケーリング機能 (Scaling) を OFF にする、という仕様に基づき作成されたコンフリクト処理ルールである。

【 0 0 5 2 】

この例において、“\$” で始まる状態変数は一時的な状態変数、“Layout\$” で始まる状態変数は実際のプリンタ機能に結びついた状態変数であるとする。製本印刷も両面印刷もどちらも同じことを行おうとしているが、製本印刷においては一時的な状態変数を用いる点が異なっている。

【 0 0 5 3 】

ここでユーザの操作により、製本印刷が選択されたとしてコンフリクト処理を開始する。仕様から判断すると、(1)→(2)→(3)→(4)→(5)の順番でコンフリクト処理ルールが評価されることを期待して記述したコンフリクト処理ルールであることから、順を追って処理を見ていく。まず、製本印刷が選択されると、(1)のルールが成立し、一時的な状態変数 \$TEMP が OFF になる。次に、\$TEMP が OFF であることから (2) のルールが成立し、\$TEMP2 が OFF になる。そして、\$TEMP2 が OFF であることから (3) のルールが成立し \$TEMP3 が OFF になる。次に (4) のルールが評価されるが、印刷の方法は製本印刷であるため、(4) のルールは成立しない。そして (5) のルールが評価され、印刷データのスケーリング機能 (Scaling) が OFF になる。

【 0 0 5 4 】

しかし、実際の動作は推論エンジンによる状態変数の推論が行われるため、必ずしも上から順にルールが評価されるわけではない。状態変数の推論は、状態変数の値が変化したことに関連する状態変数のコンフリクトルールを全体のルールから導き出し、関連するコンフリクトルールを先行して評価することで、すべてのコンフリクト処理ルールを値が変化することに評価するような処理のオーダ (手間) を省き、無限ループに陥ることを防止するように動作する。

【 0 0 5 5 】

したがって、このルールにおいては製本印刷が選択されたときに (1)→(2)→(4)→(5)→(3) と評価することも起こりうる。この場合、(5) の評価が行われるタイミングでは、\$TEMP3 の値は OFF になっておらず、一時的な状態変数の初期値である不定値状態のままである。したがって、(5) の評価は失敗してしまい、倍率が OFF になくなってしまう。

【 0 0 5 6 】

このようなときに「wait」の機能を使用する。(5) のルールを以下のように書き換える。

【 0 0 5 7 】

```
Layout$Scaling(OFF) :- $TEMP3(OFF), wait([$TEMP3]).  (5')
```

【 0 0 5 8 】

これにより、\$TEMP3 に関連するコンフリクト処理ルール (ここでは (3) を指す) が評価される前に (5') の評価が行われても、\$TEMP3 に関連するコンフリクト処理ルールが評価

されていない間は待ち行列の最後に挿入されるため、(3)の評価が行われる前に実際に評価されることはない。

【0059】

このように、コンフリクト処理ルール同士に順序がつけられることによって、現在あるコンフリクト処理ルールに対して、ルールの追加・削除が容易に行うことができる。

【0060】

また、ユーザインターフェースの更新処理、メッセージ処理もコンフリクト処理ルールに加えることで、開発者にとって可読性の高く、メンテナンスも容易なコーディングを実現することができる。さらには、ドライバのコンフリクト処理を一部だけカスタマイズするときに、ユーザインターフェース制御部のコーディングをまったく変更することなくカスタマイズが可能という効果ももたらす。その際に本発明と組み合わせれば、推論エンジンの動作を詳細に理解することなく、コンフリクト処理ルールを容易に記述し、期待した結果が得られる効果がある。

【0061】

(他の実施形態)

以上、本発明の実施形態を詳述したが、本発明は、例えばシステム、装置、方法、プログラムもしくは記憶媒体等としての実施態様をとることが可能である。また、本発明は、複数の機器から構成されるシステムに適用してもよいし、また、一つの機器からなる装置に適用してもよい。

【0062】

なお、本発明は、前述した実施形態の機能を実現するソフトウェアのプログラムを、システムあるいは装置に直接あるいは遠隔から供給し、そのシステムあるいは装置のコンピュータがその供給されたプログラムコードを読み出して実行することによっても実現される。その場合、プログラムの機能を有していれば、その形態はプログラムである必要はない。

【0063】

従って、本発明の機能処理をコンピュータで実現するために、そのコンピュータにインストールされるプログラムコード自体も本発明を実現するものである。つまり、本発明の特許請求の範囲には、本発明の機能処理を実現するためのコンピュータプログラム自体も含まれる。

【0064】

その場合、プログラムの機能を有していれば、オブジェクトコード、インタプリタにより実行されるプログラム、OSに供給するスクリプトデータ等、プログラムの形態を問わない。

【0065】

プログラムを供給するための記録媒体としては、例えば、フレキシブルディスク、ハードディスク、光ディスク、光磁気ディスク、MO、CD-ROM、CD-R、CD-RW、磁気テープ、不揮発性のメモ리카ード、ROM、DVD (DVD-ROM, DVD-R) などがある。

【0066】

その他、プログラムの供給方法としては、クライアントコンピュータのブラウザを用いてインターネットのホームページに接続し、そのホームページから本発明のコンピュータプログラムそのもの、もしくは圧縮され自動インストール機能を含むファイルをハードディスク等の記録媒体にダウンロードすることによっても供給できる。また、本発明のプログラムを構成するプログラムコードを複数のファイルに分割し、それぞれのファイルを異なるホームページからダウンロードすることによっても実現可能である。つまり、本発明の機能処理をコンピュータで実現するためのプログラムファイルを複数のユーザに対してダウンロードさせるWWWサーバも、本発明に含まれるものである。

【0067】

また、本発明のプログラムを暗号化してCD-ROM等の記憶媒体に格納してユーザに

配布し、所定の条件をクリアしたユーザに対し、インターネットを介してホームページから暗号化を解く鍵情報をダウンロードさせ、その鍵情報を使用することにより暗号化されたプログラムを実行してコンピュータにインストールさせて実現することも可能である。

【0068】

また、コンピュータが、読み出したプログラムを実行することによって、前述した実施形態の機能が実現される他、そのプログラムの指示に基づき、コンピュータ上で稼動しているOSなどが、実際の処理の一部または全部を行い、その処理によっても前述した実施形態の機能が実現され得る。

【0069】

さらに、記録媒体から読み出されたプログラムが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書き込まれた後、そのプログラムの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるCPUなどが実際の処理の一部または全部を行い、その処理によっても前述した実施形態の機能が実現される。

【図面の簡単な説明】

【0070】

【図1】実施形態における印刷処理システムのブロック構成図である。

【図2】実施形態におけるプリンタドライバUI制御モジュールの機能構成を示す図である。

【図3】実施形態におけるコンフリクト処理に係るデータの関連を示す例である。

【図4】実施形態におけるコンフリクト処理ルールの適用手順を示すフローチャートである。

【図5】ルールの適用順序に係る制御コマンドを含まないコンフリクト処理ルールの記述例を示す図である。

【図6】ルールの適用順序に係る制御コマンドを含んだコンフリクト処理ルールの記述例を示す図である。

【図7】実施形態におけるプリンタドライバUIの一例を示す図である。

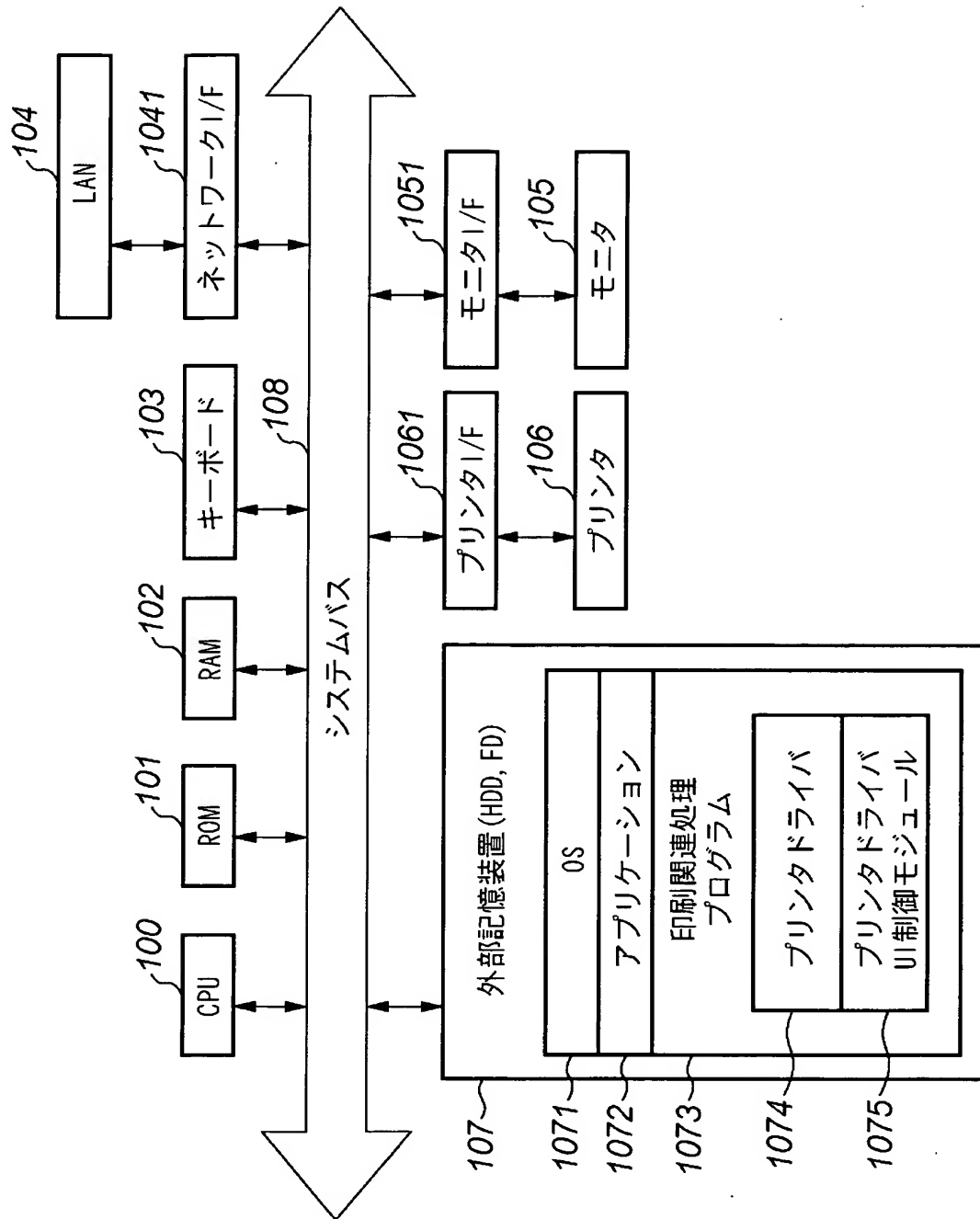
【図8】スコープを適用したコンフリクト処理ルールの記述例を示す図である。

【図9】実施形態におけるコンフリクト処理ルール記述ファイルの読み込み処理を示すフローチャートである。

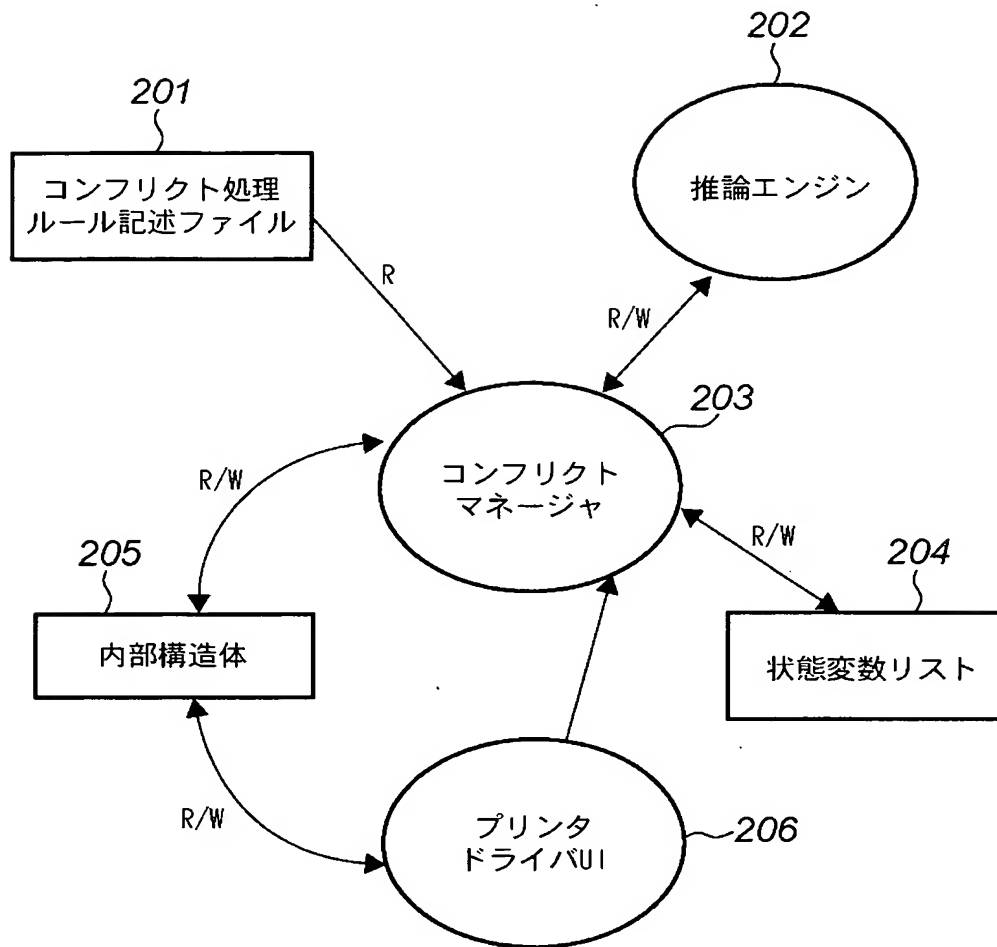
【図10】ルールの適用順序に係る制御記号の効果を説明するための図である。

【図11】ルールの適用順序に係る制御記号の効果を説明するための図である。

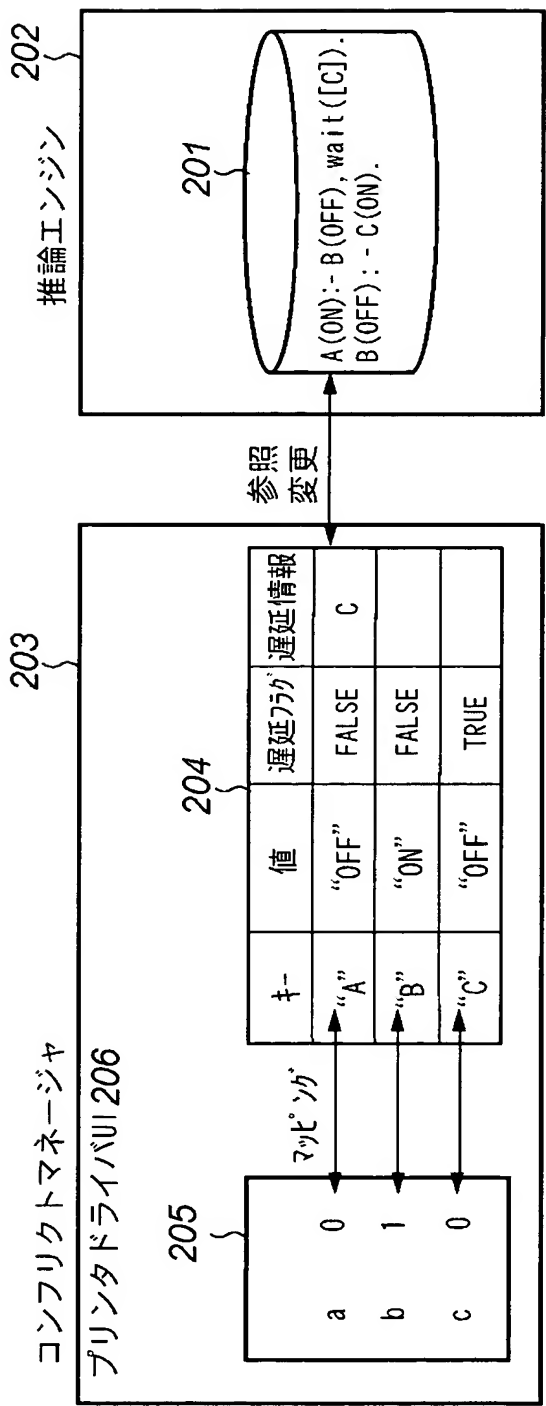
【書類名】 図面
【図 1】



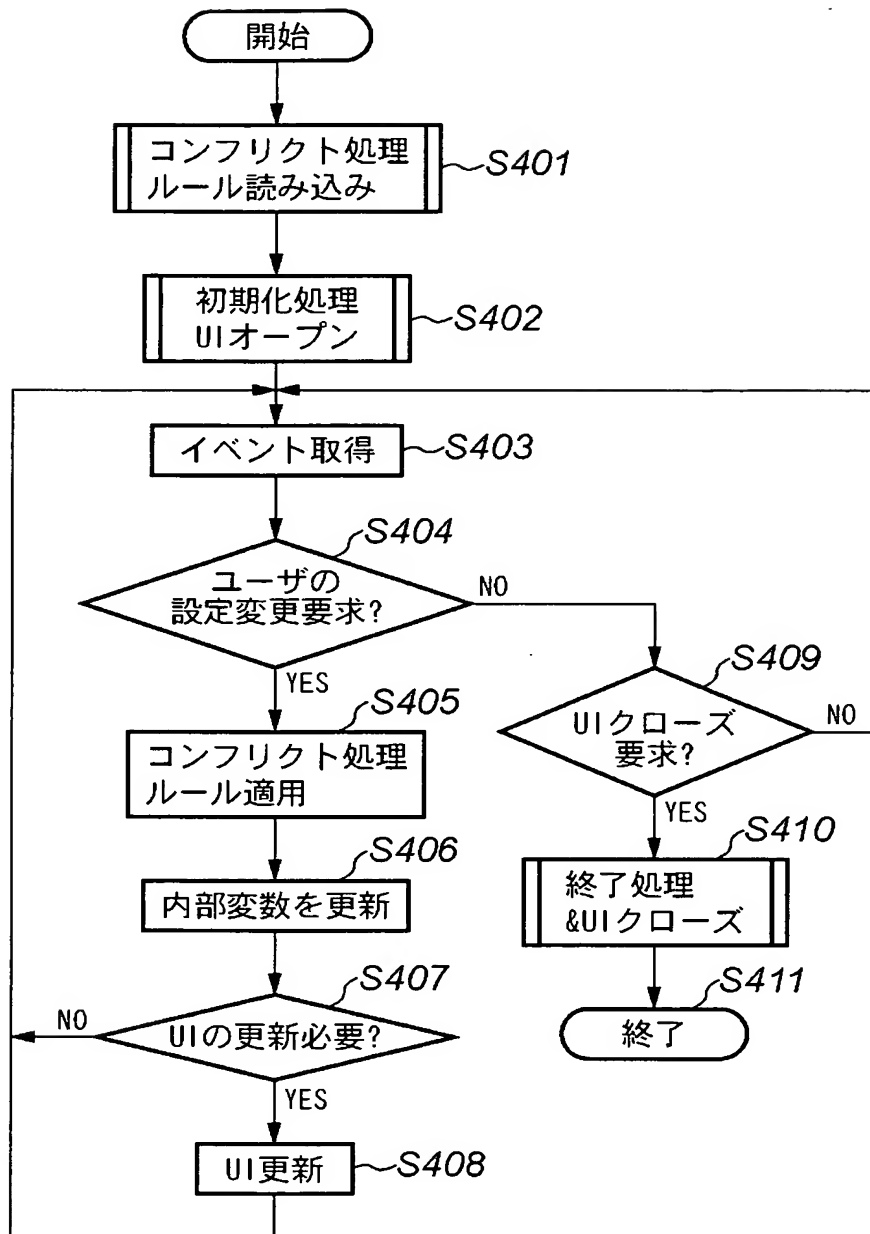
【図 2】



【図 3】



【図 4】



【図 5】

A (ON) :- B (OFF). (501)

B (OFF) :- C (ON). (502)

【図 6】

A (ON) :- B (OFF), wait([C]). (601)

B (OFF) :- C (ON). (602)

【図 7】

ページ設定 仕上げ 給紙 印刷品質

お気に入り (F): 標準設定

印刷方法 (Y):

製本印刷

製本詳細 (K)...

☒ 中とじ (I)

とじ方向 (B):

長辺とじ (左)

とじ代指定 (U)...

排紙方法 (H):

☐ 指定しない ☐ シフト (E)

☒ ソート ☐ 回転 (Q)

☐ グループ

☐ スタイプル

☐ 強制的に排紙先を固定する (I)

排紙方法 (H):

スタイプル位置指定 (L)...

仕上げ詳細 (S)...

標準に戻す (R)

OK キャンセル 適用 (A) ヘルプ

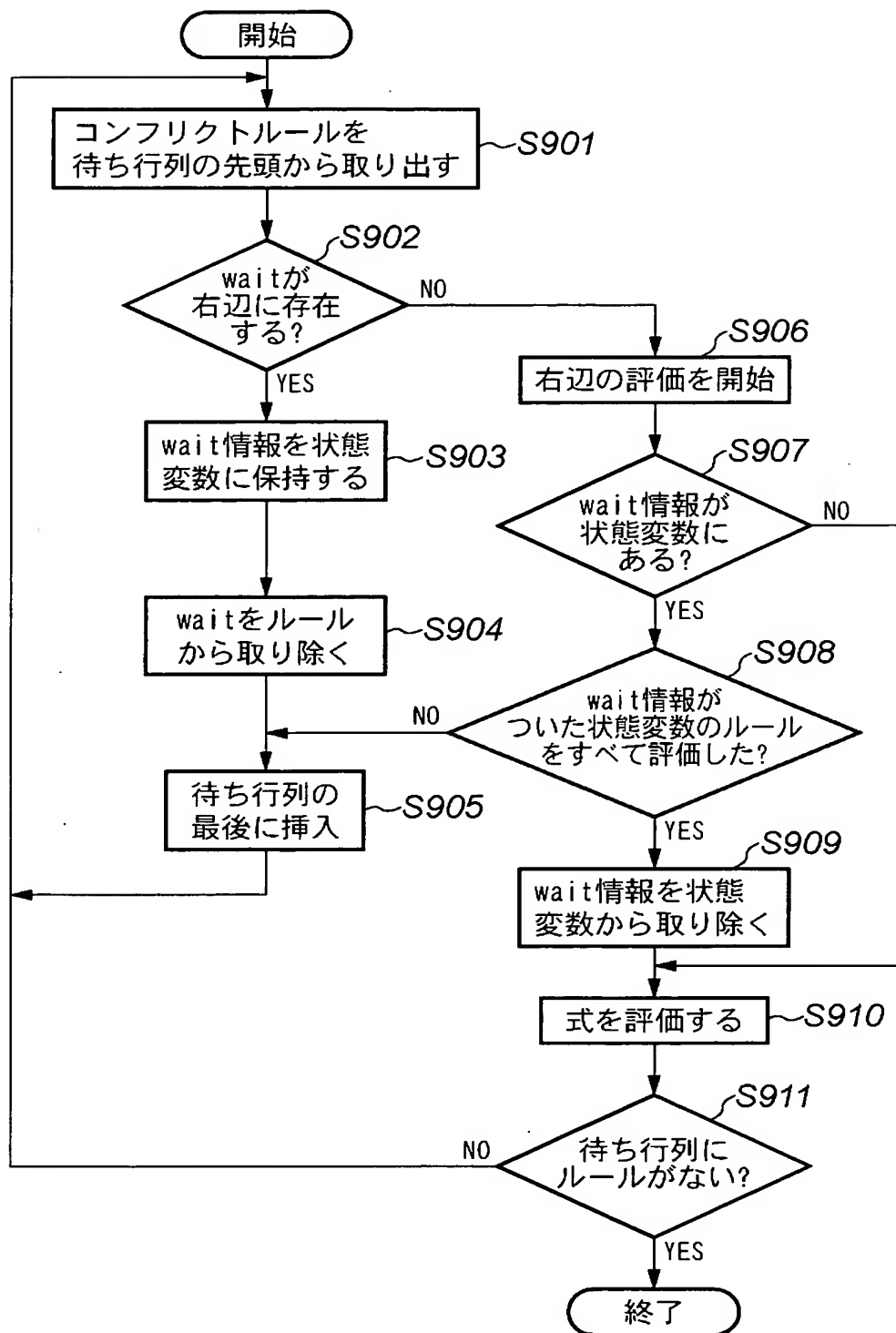
【図 8】

```

{
  B (OFF) :- C (ON).
  A (ON) :- B (OFF).
}.
{
  B (ON) :- D (ON).
  A (OFF) :- B (ON).
}.

```

【図 9】



【図 10】

A (ON) :- B (OFF).
 B (OFF) :- C (ON).
 A (ON) :- B (ON).
 B (ON) :- D (ON).

●

【図 1 1】

A (ON) :- B (OFF), wait. ([C]).

B (OFF) :- C (ON).

A (ON) :- B (ON), wait ([C]).

B (ON) :- D (ON).

【書類名】 要約書

【要約】

【課題】 開発者のミスや作業工数を減らすことができるよう、ユーザインターフェースのコンフリクト処理ルールの記事構造を改良する。

【解決手段】 デバイスドライバに設定される設定値を制御するための情報処理装置および方法において、デバイスドライバの設定値間の不整合を回避するための、複数のコンフリクト処理ルールを読み込み、読み込んだ各コンフリクト処理ルールに従って、対応する設定値の状態を制御する。ここで、上記複数のコンフリクト処理ルールの少なくともいずれかは、他のコンフリクト処理ルールに対する適用優先度の情報を含む。

【選択図】 図 4

認定・付加情報

特許出願の番号	特願 2 0 0 4 - 0 7 4 7 0 6
受付番号	5 0 4 0 0 4 3 2 9 8 7
書類名	特許願
担当官	第七担当上席 0 0 9 6
作成日	平成 1 6 年 3 月 1 9 日

< 認定情報・付加情報 >

【特許出願人】

【識別番号】	000001007
【住所又は居所】	東京都大田区下丸子 3 丁目 3 0 番 2 号
【氏名又は名称】	キヤノン株式会社

【代理人】

申請人	
【識別番号】	100076428
【住所又は居所】	東京都千代田区紀尾井町 3 番 6 号 秀和紀尾井町 パークビル 7 F 大塚国際特許事務所
【氏名又は名称】	大塚 康德

【選任した代理人】

【識別番号】	100115071
【住所又は居所】	東京都千代田区紀尾井町 3 番 6 号 秀和紀尾井町 パークビル 7 F 大塚国際特許事務所
【氏名又は名称】	大塚 康弘

【選任した代理人】

【識別番号】	100112508
【住所又は居所】	東京都千代田区紀尾井町 3 番 6 号 秀和紀尾井町 パークビル 7 F 大塚国際特許事務所
【氏名又は名称】	高柳 司郎

【選任した代理人】

【識別番号】	100116894
【住所又は居所】	東京都千代田区紀尾井町 3 番 6 号 秀和紀尾井町 パークビル 7 F 大塚国際特許事務所
【氏名又は名称】	木村 秀二

特願 2 0 0 4 - 0 7 4 7 0 6

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 1 0 0 7]

1. 変更年月日	1 9 9 0 年 8 月 3 0 日
[変更理由]	新規登録
住 所	東京都大田区下丸子 3 丁目 3 0 番 2 号
氏 名	キャノン株式会社